AN HEURISTIC SCHEDULING ALGORITHM FOR RESOURCE-CONSTRAINED PROJECT NETWORKS

Stewart Iden Marsh

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

An Heuristic Scheduling Algorithm
for
Resource-Constrained Project Networks

by

Stewart Iden Marsh, Jr.

December 1976

Thesis Advisor: A. W. McMasters

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
An Heuristic Scheduling Algorithm for Resource-Constrained Project Networks

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis
December 1976

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Stewart Iden Marsh, Jr.

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, CA 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, CA 93940

12. REPORT DATE
December 1976

13. NUMBER OF PAGES
40

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Naval Postgraduate School
Monterey, CA 93940

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Scheduling
Networks
Heuristic Scheduling

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
An algorithm is proposed for scheduling project networks having a single constraining resource and a constant level of available resources. The algorithm seeks to generate a minimum length schedule indirectly by maximizing the average resource utilization over the two time intervals represented by the current decision point and its successor. An attempt

20. Abstract (continued)

is made to schedule all of the activities whose predecessors have been completed, failing this all feasible subsets are considered. Where possible, the algorithm considers only those subsets which introduce new activities at the subsequent decision point.

An Heuristic Scheduling Algorithm for Resource-Constrained Project Networks

by

Stewart Iden Marsh, Jr.
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
December 1976

ABSTRACT

An algorithm is proposed for scheduling project networks having a single constraining resource and a constant level of available resources. The algorithm seeks to generate a minimum length schedule indirectly by maximizing the average resource utilization over the two time intervals represented by the current decision point and its successor. An attempt is made to schedule all of the activities whose predecessors have been completed, failing this all feasible subsets are considered. Where possible, the algorithm considers only those subsets which introduce new activities at the subsequent decision point.

TABLE OF CONTENTS

LIST OF DRAWINGS

TABLE OF SYMBOLS

$G(N,E)$	a general project network
N	the set of nodes in $G(N,E)$
E	the set of arcs in $G(N,E)$
(i,j)	an individual arc, a member of E representing an activity
d_{ij}	the duration of activity (i,j)
r_{ij}	the resource requirement of activity (i,j)
LS_{ij}	the latest start time for activity (i,j)
A	the quantity of resources available for use by project activities
T	an index of the elapsed time since the project start
Y_T	the eligible set at T; the set of unscheduled activities all of whose predecessors have been completed by T
S_j	the jth maximal subset of Y_T
t_o	the earliest completion time of an activity if S_j were scheduled to begin at T
$Y^j_{t_o}$	the eligible set at t^j_o, conditional on scheduling S_j at T
B	a subgroup of Y_T containing those S_j's whose $Y^j_{t_o}$ is a proper subset of S_j
C	a subgroup of Y_T containing all members not in B
S^*_j	the subset selected for scheduling
CN	a sub-category of B or C which contains those S_j's whose $Y^j_{t_o}$ requires more than the resources to be available at t_o
CY	a sub-category of B or C which contains those S_j's whose $Y^j_{t_o}$ requires no more than the resources to be available at t_o
t_o	the earliest completion time of an activity
Z	a dummy variable which can equal either B or C

I. INTRODUCTION

A. PAST WORK

There exists an extensive literature concerned with project scheduling in its various forms. References 1 and 6 provide basic expositions on the procedures and bibliographies on network planning and scheduling techniques. A large portion of this literature deals with the Critical Path Method (CPM) and extensions of it. Within its assumptions, CPM provides an optimum solution to the problem of finding a feasible schedule of shortest length. The use of CPM requires the assumption that the various project activities are independent to the extent that, within the limits defined by precedence relationships, activities can occur simultaneously with all other activities. In the case of limited resource availability, this assumption does not necessarily hold and the procedure is not appropriate.

Solution procedures for the case in which resources are restricted can be segregated into two basic categories. Exact methods search for an optimum solution but may require great computational effort. Heuristic procedures seek good, not necessarily optimum, solutions while minimizing computation.

There are two basic types of exact procedures. The problem could be formulated as an integer program [1], with an objective of minimizing a function reflecting schedule

length and constraints concerning resource requirements, precedence relationships and activity continuity. A formulation for a moderate sized project might require thousands of variables and constraints. For example, a project with 55 activities using 4 types of resources with a time span of 30 days has 6,870 constraints and 1,650 variables (not counting slack variables or the additional equations and variables necessary to assure an integer solution [1]).

Alternatively, an enumeration method [2], [3] might be used. These, in effect, determine and consider all possible feasible solutions and select one with the minimum length. Even using successively smaller bounds to eliminate obviously poor solutions, a moderately sized project might consider hundreds or thousands of alternatives.

Heuristic methods for resource allocation sacrifice the guarantee of an optimum solution by considering a small subset of all feasible solutions. The most popular algorithms are found in references 1 and 6. The particular subset of feasible schedules considered is determined by the choice of a priority rule for selecting activities for scheduling of which there are numerous varieties in open literature. Davis and Patterson [5] describe and compare seven of these with the optimum with respect to their performance on 83 projects consisting of 20 to 27 activities. Their results indicated that rules based on activity slack, activity late-finish-time or a measure of activity delay had the best performance as measured by the average percentage increase in schedule length

over the optimum length. Their average increases ranging from 5.6 to 6.8 percent. The penalties associated with using heuristic methods were also indicated by their results which show that scheduling according to minimum activity slack produced the smallest mean increase (5.6 percent) with the third smallest standard deviation (6.1) and still produced 14 of 83 schedules which were more than ten percent longer than the optimum (including one which was 24 percent longer). Finally, they provide a proof of the equivalence of the schedules generated by minimum activity slack rules and minimum activity late start rules, as based on ordinary network procedures. Reference 4 provides an extensive bibliography on both exact and heuristic procedures.

B. OBJECTIVE AND SCOPE

The purpose of this paper is to present an heuristic algorithm for scheduling projects with a single constraining resource and a constant resource availability which attempts to eliminate suboptimalities produced by algorithms using minimum activity slack or late finish rules. These suboptimalities are caused by a failure to consider the consequences of simultaneously scheduling activities.

II. MODEL DEVELOPMENT

A. PROJECT DESCRIPTION

A project is considered to consist of m separate completely-identifiable activities. Each activity is characterized by three attributes:

(1) duration - the time required to perform the activity;

(2) resource requirement - the level of resource usage in each period of activity duration;

(3) immediate predecessors - a set of project activities, the completion of which must chronologically precede without intervention of other activities, the start of the subject activity.

It is assumed throughout this paper that activities once started must continue without interruption until their completion. Resources are assumed to be of a single type and homogeneous in quality. An activity's resource demand is assumed to be constant for its duration.

From the precedence relationships it is possible to deduce for each activity, a set of immediate successors. The members of this set can be started no sooner then the time of the activity's completion. All activities without predecessors are called starter activities and all activities without successors are called final activities.

B. NETWORK FORMULATION

A project may be represented by a connected network G(N,E), where N is a set of nodes and E is a set of arcs. Let the set of integers i=0,1,...,n represent the nodes and the two-tuples (i,j) (i=0,1,...,n-1; j=1,2,...,n; $i \neq j$) represent the arcs. The node 0 corresponds to the network's initial node and node n corresponds to its terminal node. Arcs are assumed to be directed from node i, called the initial node, to node j, called the terminal node. It is also assumed, for notational ease, that if there exists an arc between nodes i and j, regardless of orientation, then it is unique. If initially a conflict occurs, an additional node k can be created and made the terminal node of one of the arcs. Thus (i,j) becomes (i,k), and an arc (k,j), called a dummy, should then be created to maintain the precedence relation.

In a network which correctly represents a particular project, there exists a one-to-one correspondence between the project's activities and the network's arcs according to the following rules.

(1) All starter activities have 0 as their initial node; i.e., they are of the form (0,j).

(2) All final activities have n as their terminal node; i.e., they are of the form (i,n).

(3) All other activities are of the form (i,j), where their immediate predecessors are (a,i), and their immediate successors are (j,b).

(4) Each arc is characterized by the duration (d_{ij}) and resource requirement (r_{ij}) of its associated activity.

(5) Dummy arcs have zero durations and resource requirements.

C. DEFINITION OF NETWORK CRITICAL PATH AND ACTIVITY LATE START TIMES

Consider the maximum duration tree rooted at node 0 of the project network G(N,E). Associate with each node i a value ES_i which is the sum of the durations of the activities lying on the path between node 0 and node i. This value represents the earliest time after the start of the project that it will be technologically possible to start activities having i as an initial node. Thus ES_n is the smallest amount of time in which it is technologically possible to perform the entire project. The path from node 0 to node n which generates ES_n is called the critical path.

Now consider the maximal duration tree rooted at n. Associate with each node j a value L_j where $L_n=ES_n$ and, for $j\neq n$, $L_j=L_n-l_{jn}$ (l_{jn} is the length of the longest path between nodes j and n). Assign to each arc (i,j) a value $LS_{ij}=L_j-d_{ij}$, which is the latest time that activity (i,j) can be started without extending the length of the schedule. Moder and Phillips [6] provide an algorithm to determine these values.

Figure 1 represents a typical arc with its associated characteristics, predecessors and successors. Solid arrows correspond to project activities and the dashed arrow is a dummy arc.

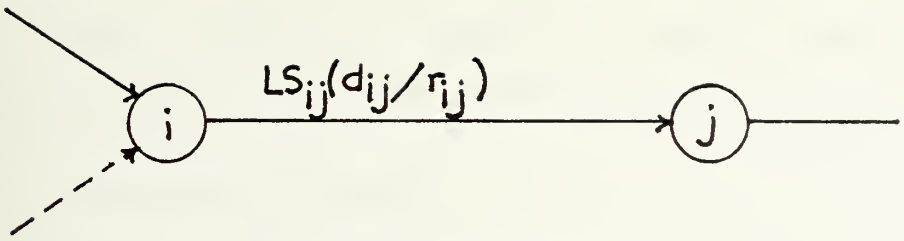

Figure 1. Representative arc

D. RESOURCE AVAILABILITY

Associated with the project is a value (A) which represents the amount of the single scarce homogeneous resource available in each time period. Throughout this paper it is assumed that this value is constant. With this assumption, a necessary condition for the existence of a feasible schedule is that $A \geq \max \{r_{ij}\}$.

E. PROJECT SCHEDULES

A project schedule is defined to consist of a set of assigned start times for each of the activities in the project. It is assumed that the project is started at time 0, so that F, the project completion time, is equivalent to the schedule's length. Associated with any schedule is a vector $Q = (q_1, q_2, \ldots, q_F)$, where q_t represents the level of resource usage in time period t-1,t.

A feasible schedule is defined to be a project schedule in which:

(1) all technological constraints are observed;

(2) $q_t \leq A$, for $t = 0,1,\ldots,F$;

(3) the schedule length is finite.

An optimum schedule is a feasible schedule whose length is at least as short as the length of any other feasible schedule.

F. TIME-SCALED NETWORKS

For purposes of clarity and simplicity, a project schedule will be illustrated by a device called a time-scaled network. This is a combination of a Gantt Chart, in which activities are displayed along a horizontal time scale, and a project graph depicting precedence relationships. Figure 2 is an example of a time-scaled network.

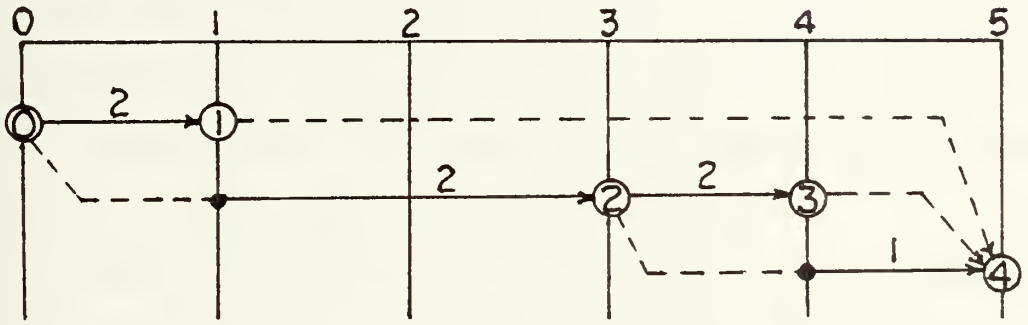

Figure 2. A time-scaled network.

The scale along the top is marked off in unit time periods. Solid arrows represent project activities with their tail at their assigned start time, their head at their completion time and their length scaled to their duration. Dashed lines indicate that an activity has slack and help maintain precedence relationships. The numerals in circles are the network nodes and are located at the earliest assigned start time of the activities for which they are an initial node. Activity starts and completions at other than project nodes are indicated by solid dots. Each arc has a number above it which represents its resource requirement.

III. SOLUTION PROCEDURE

A. INTRODUCTION

Generally, heuristic scheduling algorithms consist of three basic steps. In the initial step, any parameters used by the algorithm are set to their starting values. The second or general step determines the set of eligible activities (those whose predecessors have been completed) and then selects and schedules a subset of the eligible activities which satisfies the resource constraints. Finally, there is a bookkeeping step where the stopping conditions are tested and, if they are not satisfied, the parameters are incremented and control is returned to the general step.

The heart of an heuristic algorithm is its procedure for selecting the particular subset of eligible activities to be scheduled. At any particular time t, there are several bits of information available for consideration. These are the three attributes of each of the activities of the eligible set and of the set of activities previously scheduled and which will be completed after time t, and the late start times for the members of the eligible set. In general, the effectiveness of an algorithm is directly related to the amount of information considered and the degree of foresight exercised.

In the quest for algorithmic simplicity, a typical procedure considers,

(1) one of the numerical parameters (LS_{ij},d_{ij},r_{ij}) of the eligible set;

(2) possibly a second numerical parameter to break ties; and

(3) the previously scheduled activities only so far as they decrease the resources currently available. Rarely, if ever, is information about the successors considered. The activity late start time is the only parameter which contains any information about the future.

When the typical heuristic algorithm selects a feasible subset of the eligible set of activities, if first ranks the members of the eligible set according to its particular decision parameter. Activities are then considered one at a time in order of decreasing rank and if there are sufficient resources available it becomes a member of the subset and resources are obligated for its scheduling. This continues until either all activities have been considered or all resources have been obligated. The process has two weaknesses. First, large amounts of resources can remain idle in the current period which could be utilized if the activities were considered in a different order. Secondly, because it is shortsighted, the delayed activities may not be schedulable in the next period and will be delayed again, or savings contributed by scheduling successors with members of the current eligible set may be missed.

When a different ordering of activities would result in an increase in the total usage of resources over the two periods a suboptimality occurs. Consideration of both the eligible set and its successors would avoid this suboptimality.

Consider Figure 3, in which the use of a minimum late start rule generates such a suboptimality.

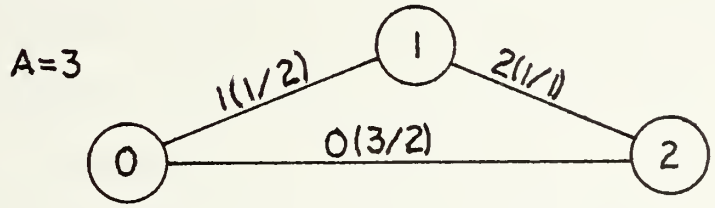

Figure 3a. The project network.

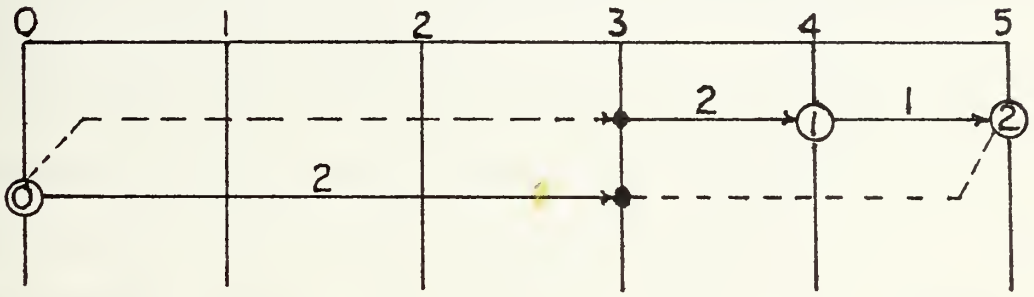

Figure 3b. Schedule generated using min LS_{ij}.

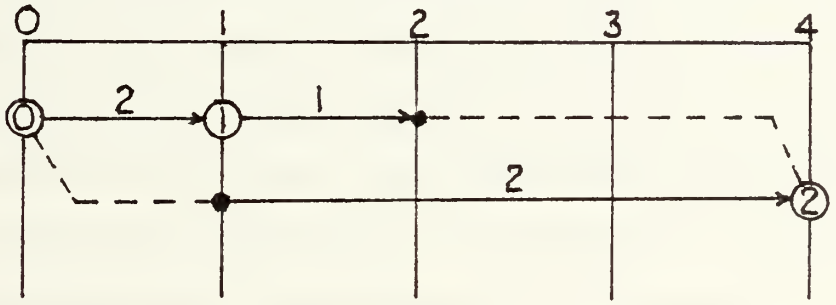

Figure 3c. Optimum schedule.

The min LS rule schedules (0,2) at T=0. Insufficient resources exist to schedule (0,1) so it is delayed, and the resulting schedule is five units long as shown in Figure 3b. Figure 3c shows the results of scheduling (0,1) first. It is obvious that no activity can occur simultaneously with (0,1). However, activities (0,2) and (1,2) can function concurrently. A schedule of length four results.

B. PREVIEW

The proposed algorithm is based upon consideration of both the current set of eligible activities and the members of the eligible set at the next decision point. The algorithm attempts to generate a minimal length schedule indirectly by maximizing the average utilization of resources over the two periods of consideration.

In situations for which there are insufficient resources to schedule all of the currently eligible activities, an initial goal is to try to introduce new activities at the next decision point. This serves to arbitrarily define the successors of a feasible subset which does not add activities to the next eligible set as low priority and seeks to avoid a situation in which there are gross amounts of unused resources at the next decision point. By accepting a possibly lower level of resource usage now, it is hoped that the increase in the number of options, represented by the activities added to the next eligible set, will return a bonus in the form of an increase in resource usage in the future and raise the average usage over the two periods.

This impact is most explicit when selecting between a final activity and one with a successor. The partial network in Figure 4 illustrates this situation. If (a,d) and (c,d) are simultaneously schedulable, then the smallest increment to the schedule's length will be achieved by scheduling (b,c)

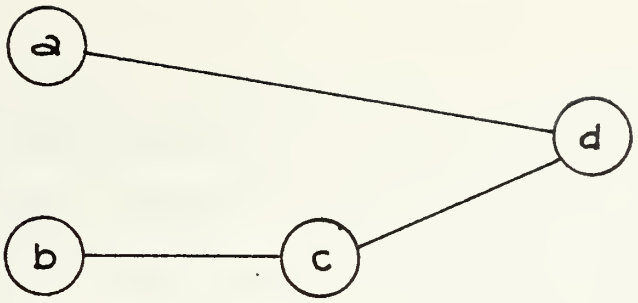

Figure 4. Project termination case.

first, resulting in an increment of $d_{bc} + \max(d_{ad}, d_{cd})$. The alternative of scheduling (a,d) first contributes an increment of length $d_{ad} + d_{bc} + d_{cd}$ which is necessarily longer if none of the activities have zero duration. In the case in which (a,d) and (c,d) are not simultaneously schedulable, the increment contributed by scheduling either (a,d) or (b,c) first is $d_{ad} + d_{bc} + d_{cd}$ and the specific decision is inconsequential. Therefore, given that a final activity and an activity with a successor cannot be concurrently active, the shorter schedule will always be achievable by scheduling first the activity with the successor.

A secondary goal of the algorithm is to schedule activities which require a large amount of the available resources as soon as possible, rather than as many activities as is possible at each step.

Consider an example in which the eligible but infeasible set consists of four activities which can be divided into two subsets, each of which is feasible. Suppose one subset contains a single activity which requires all of the available resources in any period, and the second consists of the other three activities. The single activity set is going to have

to be scheduled by itself sometime. If it is scheduled as early as possible then the rest of the network will be delayed until its completion. Scheduling the three-activity set first would delay the single activity at least as long as the duration of the longest of the three activities. The potential exists that resources not needed by the second set will remain idle rather than being used by the single activity's successors until a subsequent time period when the single activity can again be scheduled. The implication is that bottleneck activities should be scheduled as soon as possible; the algorithm tries to do this. The exception occurs when the single activity is a final activity, we have shown that it can be delayed until last without effecting the length of the generated schedule.

At a particular point in the scheduling process, when the eligible set contains m members, there are a total of 2^m different subsets of the eligible set, including the null set and the set itself. Even after eliminating those requiring more than the available resource, the number remaining may still be considerable. Therefore, arbitrarily, any subset which is again a proper subset of one of the remaining feasible subsets will be immediately dropped from further consideration. The final result is a set whose members are called maximal feasible subsets of the eligible set. These are the only subsets of the eligible set considered by the algorithm. Although it is possible to generate examples in which choosing a maximal subset leads to a suboptimal

schedule, the evidence indicating such a suboptimality unfolds deeper into the process than one step of foresight can discover.

C. THE ALGORITHM

1. Set T=0.

2. Select the set of activities (Y_T) that can begin at time T.

3. If there are sufficient resources available to schedule all of the activities in Y_T, do so. Go to Step 6. Otherwise, examine all subsets of Y_T for which sufficient resources are available, and for which it is not possible to include another member of Y_T without exceeding the available resources. Call these maximal subsets: $S_1, S_2, \ldots, S_k$. If there exist no maximal feasible subsets of Y_T, go to Step 6.

4. For each of these subsets S_j, determine t_o^j, the earliest completion time of an activity, either a member of S_j or previously scheduled and not yet completed, given that S_j is scheduled at time T. At t_o^j, determine the set of activities previously scheduled and active beyond t_o^j and the set of $Y_{t_o}^j$, the set of activities that can begin at t_o^j if S_j is scheduled at time T.

5. For each subset S_j, examine its $Y_{t_o}^j$. If $Y_{t_o}^j$ is a proper subset of Y_T; place S_j in subgroup B, otherwise place S_j in subgroup C.

a. If C is empty, go to e.

b. Examine each S_j in C. If it will be possible to schedule all of the activities in $Y^j_{t_o}$ at t^j_o; place S_j in category CY. Otherwise, place S_j in category CN.

c. If CY is empty, to to d. Otherwise, select S^*_j from CY, according to the following criteria.

(1) Denote S^*_j as the subset containing the activity of shortest duration.

(2) If ties exist, of the subsets containing an activity of shortest duration, denote S^*_j as the subset containing the activity of second shortest duration. Continue this process, considering the third, fourth, etc. activity of shortest duration until either a unique subset remains or all of the activities of the subset with the fewest members have been tested.

(3) If ties still exist, denote S^*_j as the remaining subset which utilizes the most resources. Schedule all of the activities in S^*_j and $Y^j_{t_o}$. Go to Step 6.

d. Select S^*_j from CN as the one containing the activity with the greatest resource requirement. In the case of ties, use the criteria in c to select from among the tied subsets. Schedule all of the activities in S^*_j. Go to Step 6.

e. If, for some S_j in B, it will be possible to schedule all of the activities in $Y^j_{t_o}$ at t^j_o; place S_j in category CY; otherwise place S_j in category CN. Continue until all S_j in B have been categorized. Go to c.

6. If all project activities have been scheduled, STOP. Otherwise, determine t_o, the earliest completion time of all activities previously scheduled and not completed by time T. Set $T=t_o$ and go to Step 2.

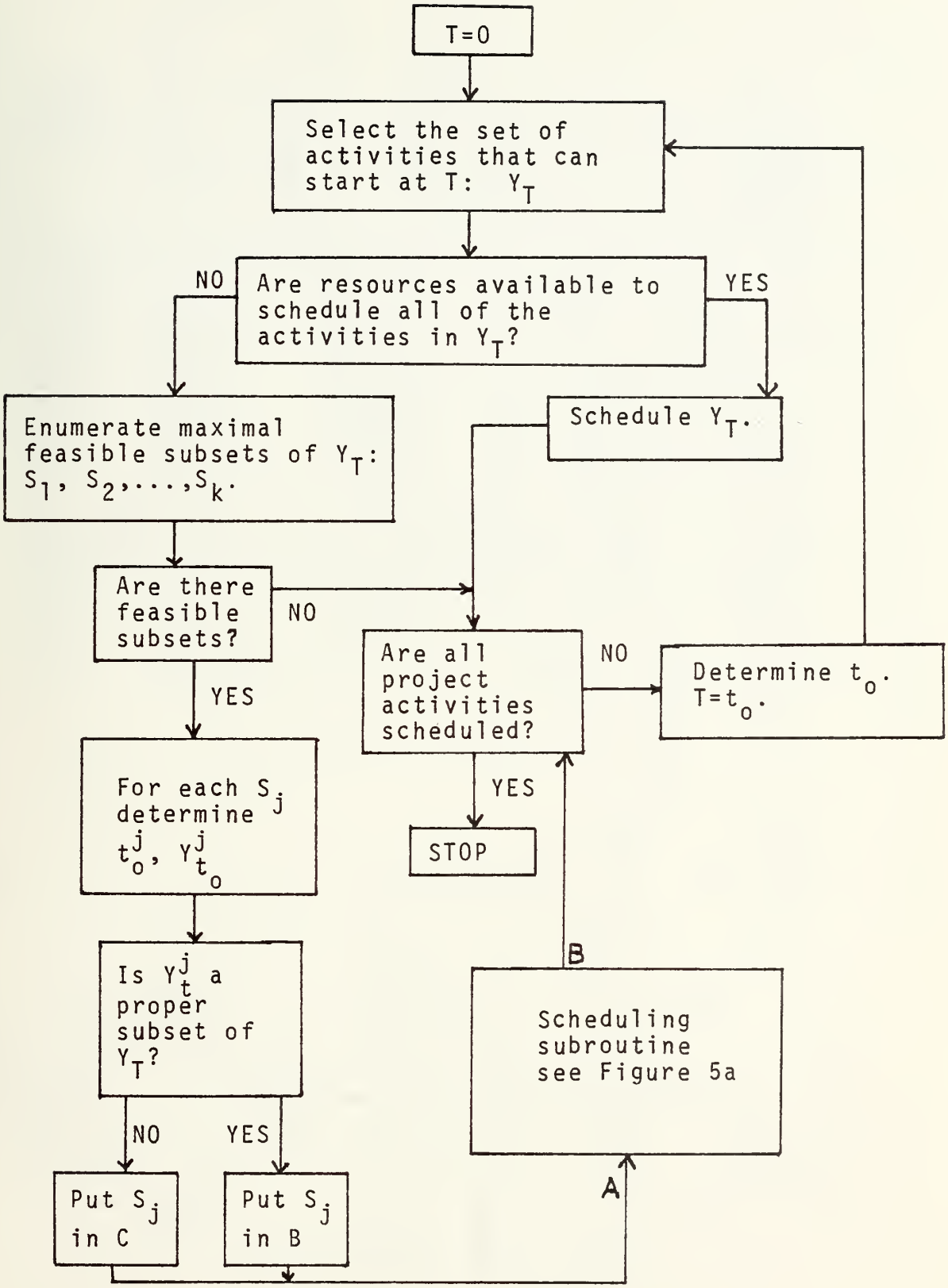

Figure 5. Flowchart of the algorithm.

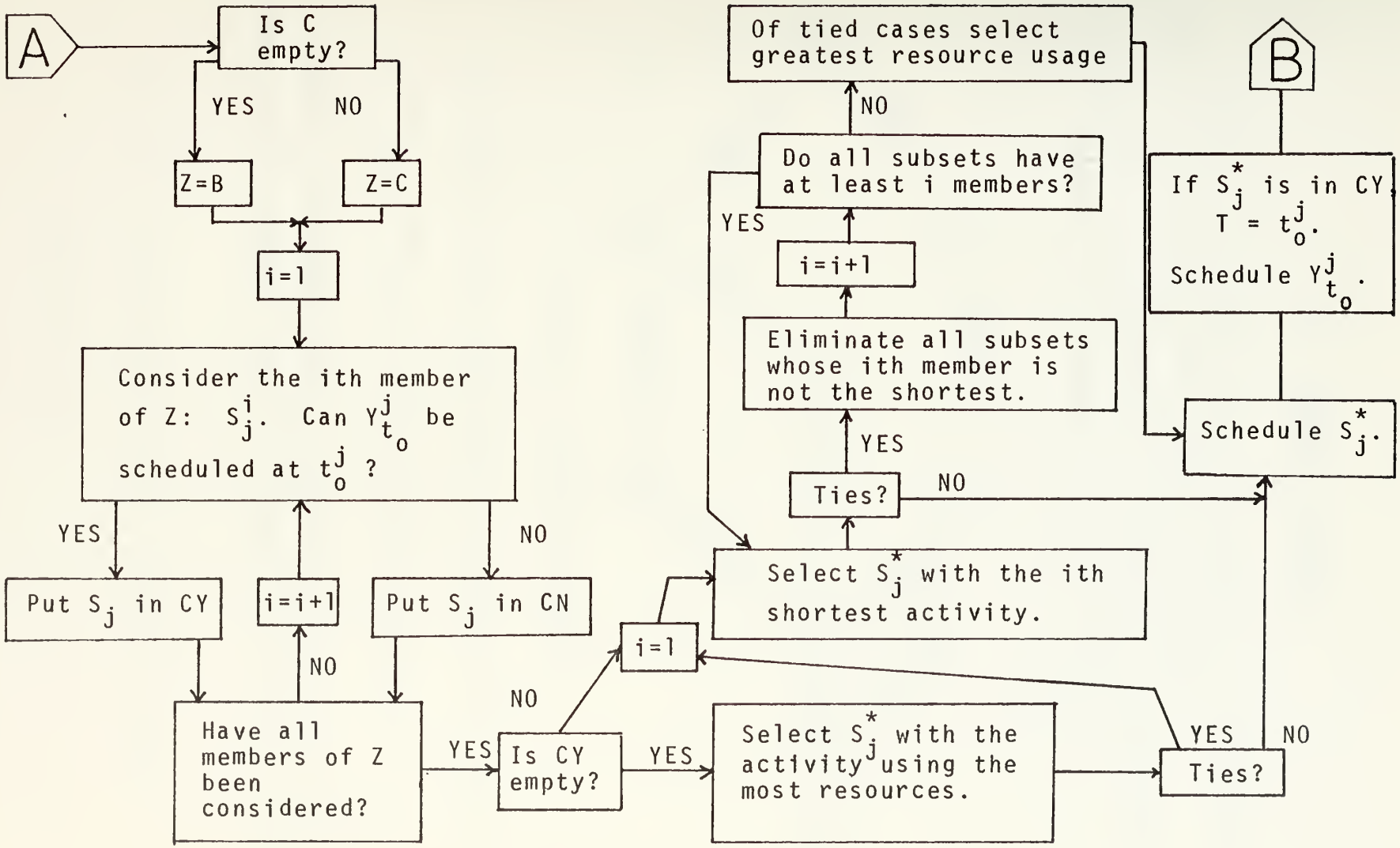

Figure 5a. Scheduling subroutine.

IV. EXAMPLE PROBLEM

Consider the network shown in Figure 6. The numerical quantities associated with each arc (i,j) are: $LS_{ij}(d_{ij}/r_{ij})$. The problem is to generate a minimum duration project schedule for a resource availability A=3.

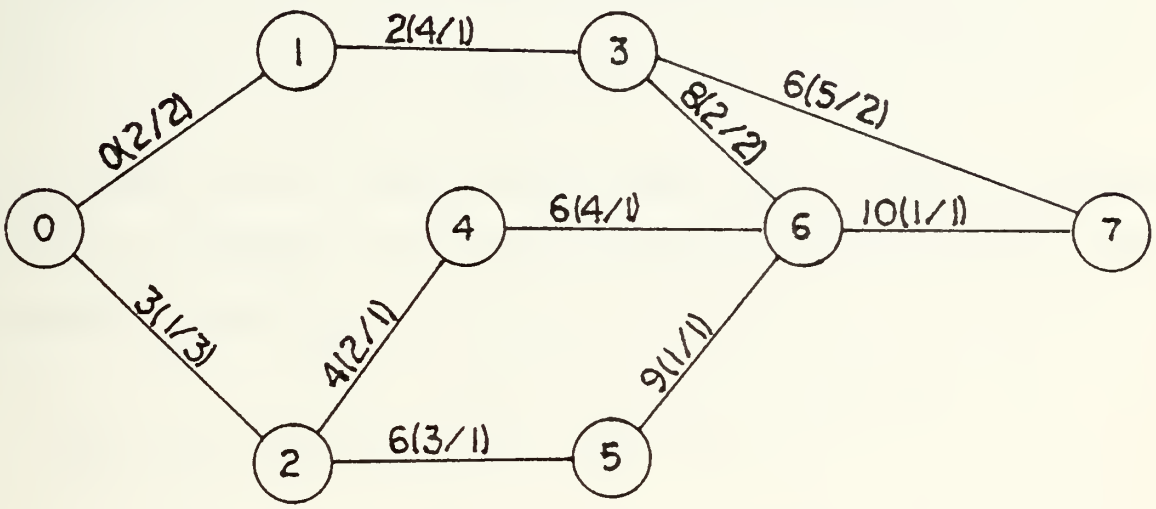

Figure 6. Example network

The steps of the algorithm for solving this problem are:

Step 1. $T=0$; $Y_0 = \{(0,1),(0,2)\}$; $\sum_{Y_0} r_{ij} = 5 > 3$.

Y_0 requires more than the available resources, therefore enumerate the maximal feasible subsets and for each determine t_o and Y_{t_o}.

(1) $S_1 = \{(0,1)\}$; $t_o^1 = 2$; $Y_2^1 = \{(0,2), (1,3)\}$;

$\Sigma r_{ij} = 4 > 3$.

(2) $S_2 = \{(0,2)\}$; $t_o^2 = 1$; $Y_1^2 = \{(0,1), (2,4), (2,5)\}$;

$\Sigma r_{ij} = 4 > 3.$

Both Y_2^1 and Y_1^2 will require more than the resources to be available, therefore schedule S_2 which contains the activity requiring the most resources, $t_o = 1 = T$.

Step 2. $T = 1$; $Y_1 = \{(0,1), (2,4), (2,5)\}$;

$$\sum_{Y_0} r_{ij} = 4 > 3.$$

The eligible set Y_1 requires more than the available resources, therefore enumerate the maximal feasible subsets and for each determine t_o and Y_{t_o}.

(1) $S_1 = \{(0,1), (2,4)\}$; $t_o^1 = 3$;

$Y_3^1 = \{(1,3), (4,6), (2,5)\}$; $\Sigma r_{ij} = 3.$

(2) $S_2 = \{(0,1), (2,5)\}$; $t_o^2 = 3$;

$Y_3^2 = \{(1,3), (2,4)\}$; $\Sigma r_{ij} = 3.$

(3) $S_3 = \{(2,4), (2,5)\}$; $t_o^3 = 3$;

$Y_3^3 = \{(0,1), (4,6)\}$; $\Sigma r_{ij} = 4 > 3.$

All three subsets lead to the introduction of new activities into the decision process and both of Y_3^1 and Y_3^2 will be completely schedulable at their respective t_o. Both S_1 and S_2 contain (0,1) as the shortest activity, but since (2,4) of S_1 is shorter than (2,5) of S_2, schedule S_1. Since Y_3^1 is schedulable, schedule its activities at t_o^1. The time of the next activity completion, t_o, is 6. At T=6, activities (1,3) and (4,6) will still be active, therefore the available resources (R) will be one (1) unit.

Step 3. $T = 6$; $Y_6 = \{(5,6)\}$; $\sum_{Y_6} r_{ij} = 1 = R$.

All of the activities in Y_6 can be scheduled without exceeding the available resources. Schedule the activities in Y_6. Thus, $t_o = 7$. All previously scheduled activities are completed by T=7.

Step 4. $T = 7$; $Y_7 = \{(3,6), (3,7)\}$; $\sum_{Y_7} r_{ij} = 4 > 3$.

The eligible set Y_7 requires more than the available resources, therefore enumerate the maximal feasible subsets and for each determine t_o and Y_{t_o}.

(1) $S_1 = \{(3,6)\}$; $t_o^1 = 9$; $Y_9^1 = \{(3,7), (6,7)\}$; $\Sigma r_{ij} = 3$.

(2) $S_2 = \{(3,7)\}$; $t_o^2 = 12$; $Y_{12}^2 = \{(3,6)\}$; $\Sigma r_{ij} = 2$.

In this case, Y_{12}^2 is a proper subset of Y_7 and S_2 is assigned to subgroup B. Since Y_9^1 is not a proper subset of Y_7, S_1 is assigned to subgroup C. All of the activities in Y_9^1 can be scheduled at $t_o^1 = 9$, so S_1 is placed in category CY. As the unique member of CY, schedule S_1. Schedule the activities of Y_9^1. All project activities have been scheduled, therefore STOP.

Figure 7 presents the time-scaled project network for the generated schedule (which also happens to be the optimum schedule). Figure 8 is the time-scaled network for the schedule generated using the heuristic algorithm due to Brooks (in Moder and Phillips [6]) based on scheduling according to minimum LS and with minimum activity duration breaking ties.

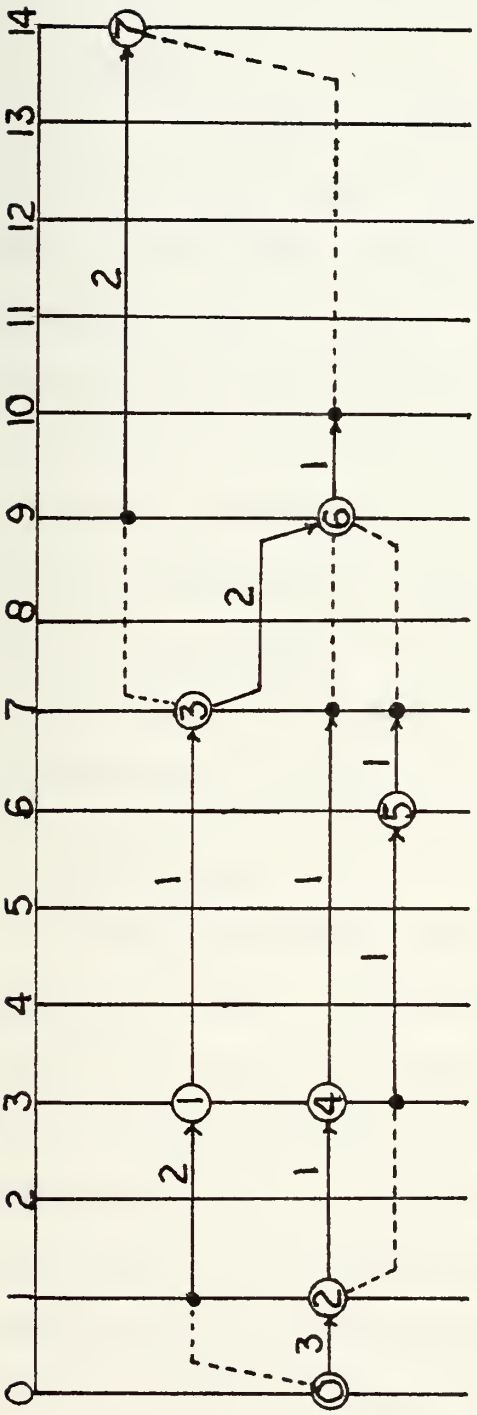

Figure 7. Schedule generated by the proposed algorithm.

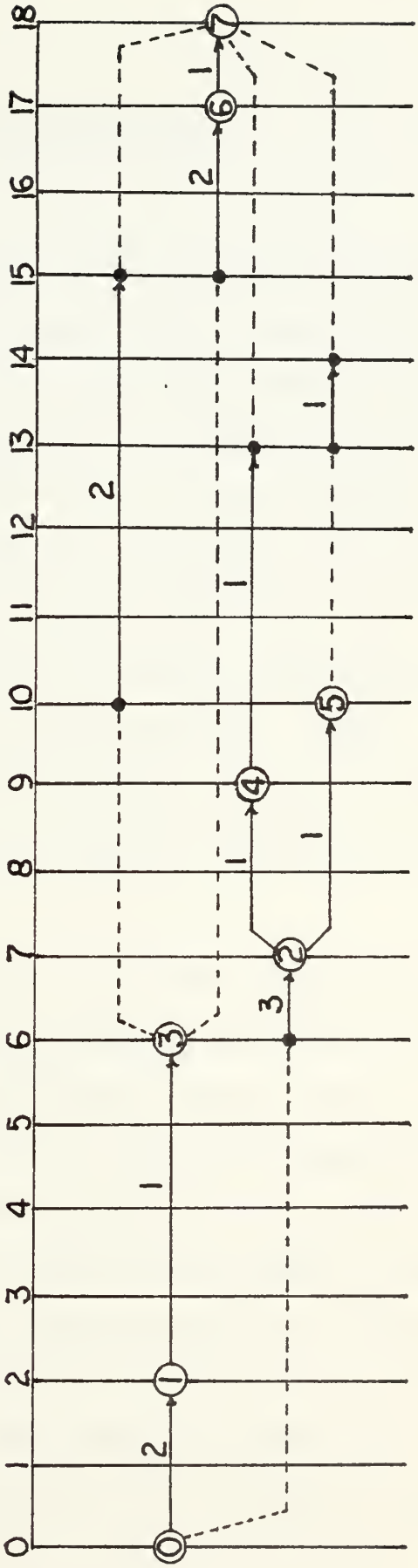

Figure 8. Schedule generated by the Brooks algorithm.

V. SUMMARY AND DISCUSSION

A. SUMMARY

An algorithm is proposed for scheduling project networks having a single type of constraining resource and a constant level of available resources. The algorithm seeks to generate a minimum length schedule indirectly by maximizing the average resource utilization over the two time intervals represented by the current decision point and its successor. An attempt is made to schedule all of the eligible activities within the resource constraints, failing this all maximal feasible subsets of the eligible activities are considered. Where possible, the algorithm considers only those subsets which introduce new activities at the subsequent decision point.

B. DISCUSSION

The simultaneous scheduling of activities becomes an important factor in the scheduling process when there is a mixture of activities which individually require most of the resources available in any period and activities requiring few resources. In these situations, the order in which activities are considered for scheduling can greatly effect the percentage of the available resources used in a period. Suppose a choice between two activities is made by a priority selection scheme such as late start times, and that an activity using only a small portion of the available resources

is selected first. If the other activity requires all of the available resources, then it cannot be scheduled until the first activity is completed. The unused resources over the duration of the first activity are effectively lost. If however, the first activity is delayed until completion of the second then it could possibly be scheduled at the same time as a successor of the second activity. By scheduling in this way the unused resources would be not more than for the previous schedule; they, hopefully, would be less. The proposed algorithm, by explicitly considering all activity orderings, seeks to avoid these suboptimalities.

The main disadvantage of using the proposed algorithm is the great increase in computational effort over a simple heuristic such as the Brooks algorithm [6]. Let m be the number of activities in the eligible set. If the set is infeasible, the number of subsets to be analyzed can vary between a lower bound of m and an upper bound of

$\frac{m!}{(\frac{m}{2})!\ (\frac{m}{2})!}$ when m is even, and $\frac{m!}{(\frac{m+1}{2})!\ (\frac{m-1}{2})!}$, when m is odd.

These upper limits occur when every activity requires exactly one unit of the resource and there are sufficient resources to schedule half of the activities. Here the simultaneous scheduling of activities has no impact on the generated schedule length since any ordering of activities will permit complete use of the available resources. The speed with which an algorithm generates a schedule is directly proportional to the number of subsets it considers at each iteration. The

Brooks algorithm considers only one subset at each step, so at best the proposed algorithm is $\bar{m}$, the average number of activities in an eligible set, times slower.

VI. EXTENSIONS AND AREAS FOR FURTHER STUDY

A. MULTIPLE TYPES OF CONSTRAINING RESOURCES

In its current form, the algorithm could be used to schedule projects with more than one type of constraining resource. A problem arises when no successor set is feasible. In this case the secondary criteria specifies that the subset containing the activity which requires the most resources be scheduled. When there is more than one constraining resource, this does not necessarily accomplish its intended objective of scheduling bottleneck activities as early as possible. For example, consider a situation in which there are two maximal feasible subsets which are identical except for one activity. The first subset contains an activity which requires two units of a resource of which there are always ten units available. The other subset contains an activity which requires one unit of this same resource and one unit of a resource of which there is only one unit ever available. All other activities require one unit of the first resource. The rule would select the first subset for scheduling because it had an activity which required two units of a resource but the second activity is clearly the bottleneck.

Modifying the rule to schedule the subset containing the activity requiring the highest percentage of an available resource again fails to achieve the results because of the possibility of different relative resource scarcities.

Consider an example in which there are two types of constaining resources. One resource, with five units available, is extremely scarce such that period after period there are insufficient resources to schedule all of the eligible activities. The other resource, with two units always available, is relatively plentiful because only two activities in the entire project require it (it is constraining because these two activities cannot be scheduled simultaneously). Using the modified rule, a subset containing an activity which required both of the available two units of the plentiful resource and none of the other would be selected over another subset containing an activity which required only four units of the scarce resource and one of the other. The activity using the scarce resource is the true bottleneck activity in this situation even though it requires a smaller percentage of its respective resources. Additional study is required to resolve these types of conflicts.

B. VARIABLE RESOURCE AVAILABILITY PROFILES

The algorithm as it stands will handle the case of variable resource profiles. The efficiency of the results may be enhanced if additional consideration is given to the implications of particular profile shapes. For example, if the level of available resources is monotonically increasing it may be advantageous to delay scheduling activities which require a large amount of resources.

C. COMPOUND STRATEGIES

The advantageous characteristic of the Brooks algorithm is the speed with which it generates a feasible schedule. But when it encounters an activity which requires a large portion of the available resources it tends to delay the activity, possibly several times. This can result in the final parts of the schedule being a sequence of single activities only constrained by precedence relationships.

A procedure combining the proposed algorithm with the Brooks algorithm could avoid such a suboptimal situation without as great an increase in computational effort as would be incurred by using the proposed algorithm by itself. First, the Brooks algorithm would be used to generate a feasible schedule. This schedule would be examined from its start for a time interval in which there are large amounts of idle resources. At this point the proposed algorithm could be applied for one iteration. The Brooks algorithm would then be reapplied to the remaining unscheduled activities and the resulting schedule reexamined from the previous stopping point for another time interval of idle resources.

D. ALGORITHM TESTING

Schedules generated by the proposed algorithm and any combined strategy should be compared with optimal schedules for a set of general project networks as was done by Davis and Patterson [5] for other heuristic algorithms. The objective

would be to determine performance measures such as the average percentage increase in schedule length and the variability in results to be expected when using the procedures.

BIBLIOGRAPHY

1. Wiest, J. D. and Levy, F. K., A Management Guide to PERT/CPM, Prentice-Hall, 1969.

2. Davis, E. W. and Heidorn, G. E., "An Algorithm for Optimal Project Scheduling Under Multiple Resource Constraints," Management Science, v. 17, No. 12, B803-B816, August 1971.

3. Shrange, L., "Solving Resource Constrained Network Problems by Implicit Enumeration - Nonpremptive Case," Operations Research, v. 18, No. 2, p. 225-235, March-April 1970.

4. Davis, E. W., "Resource Allocation in Project Network Models--A Survey," The Journal of Industrial Engineering, v. 17, No. 2, March-April 1966, p. 177-188.

5. Davis, E. W. and Patterson, J. H., "A Comparison of Heuristic and Optimum Solutions in Resource Constrained Project Scheduling," Management Science, v. 21, No. 8, April 1975, p. 944-955.

6. Moder, J. J. and Phillips, C. R., Project Management with CPM and PERT, 2nd Ed., Reinhold, 1971.

INITIAL DISTRIBUTION LIST

		No Copies
1.	Defense Documentation Center Cameron Station Alexandria, VA 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, CA 93940	2
3.	Department of Operations Research Naval Postgraduate School Monterey, CA 93940	1
4.	Commandant (G-PTP-1/72) U.S. Coast Guard 400 Seventh Street S. W. Washington, D.C. 20590	2
5.	Assoc. Professor A. W. McMasters, Code 54Mg Department of Administrative Sciences Naval Postgraduate School Monterey, CA 93940	2
6.	Assoc. Professor G. T. Howard, Code 55Hk Department of Operations Research Naval Postgraduate School Monterey, CA 93940	1
7.	LT Stewart I. Marsh Jr., USCG 4012 Trapp Road Fairfax, VA 22030	1